Bangladesh University of Engineering & Technology
Department of Computer Science & Engineering
CSE 310: Compiler Sessional
Session January 2018

Assignment 4
Intermediate Code Genaration

June 23, 2018

1 Introduction

In the previous assignment, we have performed syntax and semantic analysis of a source code written in a subset of **C** language. In this assignment you have to generate intermediate code for a source program having no error. That means if your source code does not contain any error, that was to be detected in the previous offline, you will have to generate intermediate code for the source code. We have picked 8086 assembly language as our intermediate representation.

2 Tasks

You have to complete the following tasks in this assignment.

2.1 Intermediate Code Generation

You have to generate 8086 assembly language program from the input file after the input file successfully pass all the previous steps (lexical, syntax, semantics). You have to do write codes in the same file of your previous assignment. But, you can remove the portion that calculates value. To generate assembly code you may find the following instructions helpful.

- Add a field 'code' in SymbolInfo Class.As each of your non-terminal has a SymbolInfo pointer as attribute, you can propagate code for different portion using this newly added field.
- To generate assembly code for conditional statements and loops, define two functions named **newLabel()** and **newTemp()** where newLabel will generate a new label on each call and newTemp will generate a new temporary variable.

- Write a procedure for println(ID) function and call this procedure in your assembly code whenever you reduce a rule containing println.
- Do not forget some initialization part in assembly program like initializing the data segment register in the main procedure of the generated assembly code. So you may need to check whether the **ID** in the production body of func_definition is main or not.
- You have to declare the variables declared in the source code in the data segment of the assembly code. As the variable name cab be same in different scope, you can concatenate the scope id with the variable name to make it unique.
- Handle the function call and return using stack.

  You may also find the given sample code helpful in this case. Note that the sample file consists of only some portion of the grammar. You have to generate intermediate code for all the productions given in previous grammar.txt file. This subsection may be updated. Notification will be given via moodle in such case.

2.2 Optimization

You have to do some **peephole optimization** works after intermediate code is generated. To do this you will take pair of consecutive instructions in the generated code and remove the following redundancy:

- Redundant mov instruction like

  mov ax, a
  mov a, ax

  In this case the second mov instruction can be omitted.

3 Input

The input will be a C source program in .c extension. File name will be given from command line.

4 Output

In this assignment, there will be two output file. One file, log.txt, that will contain line count, error count and any lexical, syntax or semantic errors. It would be just like your previous log file except that you don't need to print productions and symbol table. Another file, code.asm will

contain the generated assembly code before performing optimization. The last file optimizedCode.asm will contain the assembly code after optimization.

You are also encouraged to show your assembly code by running it in any emulator.

5 Submission

- Plagiarism is strongly prohibited.
- No submission after the deadline will be allowed.
- Deadline will not extend in any situation.

6 Submission

All Submission will be taken via moodle. Please follow the steps given below to submit you assignment.

1. In your local machine create a new folder which name is your 7 digit student id.
2. Put the lex file named as <your_student_id>.l and yacc file named as <your_student_id>.y containing your code. Also put additional c file or header file that is necessary to compile your lex file. Do not put the generated lex.yy.c file or executable file in this folder.
3. Compress the folder in a **zip** file which should be named as your 7 digit student id.
4. Submit the zip file within the Deadline.

7 Deadline

Submission deadline is set at **13 July, 2018**. Please start early if you want to finish the assignment.